Exhibit H

Polar codes

Develop an intuitive understanding of polar codes

EventHelix
Apr 22, 2019 · 8 min read

3GPP has selected polar codes as the error correcting code on the 5G NR control channels. Polar codes are unique in the way they split the channel into good and bad bit-channels. We will learn about the channel splitting by running experiments using Linux tools awk and gnuplot. These experiments are based on a talk by Professor Emre Telatar.

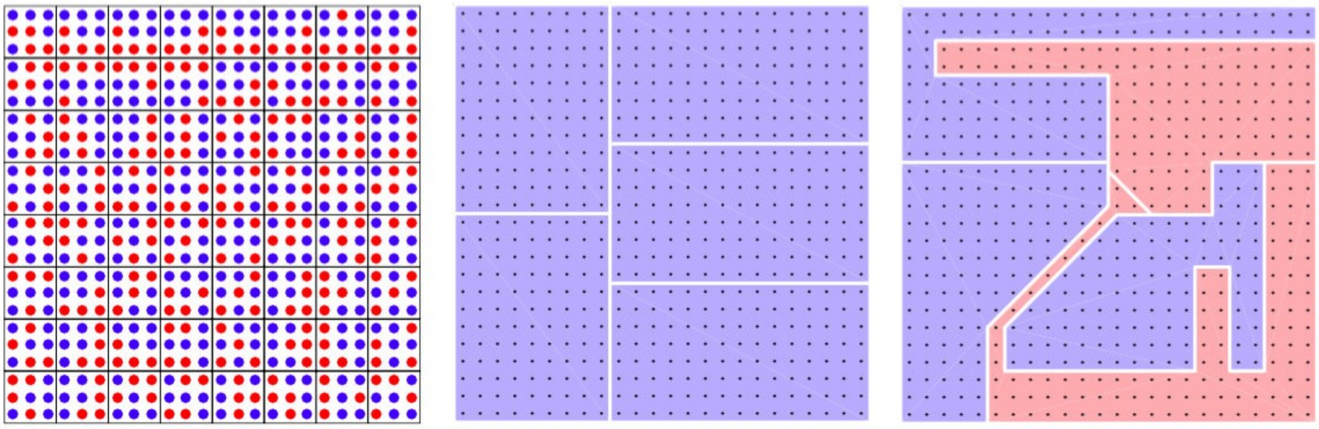
What do we mean by good and bad bit-channels? Consider a polar code where **K** information bits are being sent in a block of **N** bits. Polar code encoding will polarize the channel into reliable and unreliable bit-channels. The information bits will be transmitted on the most reliable **K** bit-channels. The remaining **N-K** channels are unreliable are usually set to 0 as they are not reliable for data transmission.

Gerrymandering

What is this process of polarization? Consider the US practice of gerrymandering illustrated in the following image:

- The rectangle on left depicts a population with 50% blue voters and 50% red voters.
- The middle rectangle represents an electoral district split the will still contain close to 50% blue and red voters in each district.
- The right rectangle shows a polarizing split that draws the electoral districts so as to make the bluer and redder.

Here we have polarized evenly distributed districts into districts that will reliably vote blue or red.

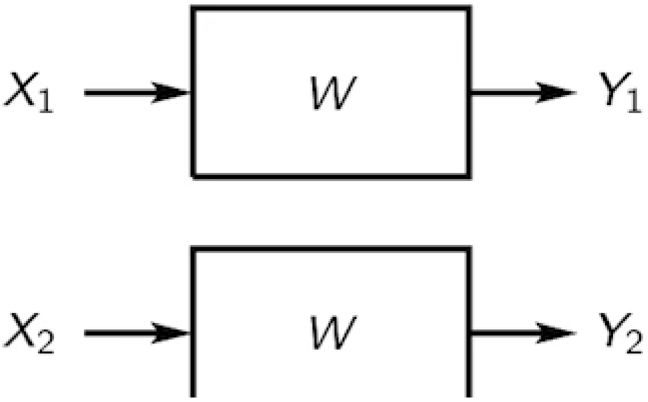


Polarizing with Gerrymandering (Credit: Phys.org)

Channel Polarization

How can we polarize a data channel into extremal good and bad channels?

Let's look at two channels shown below. Both channels implement the transform W that converts the input X to the output Y.

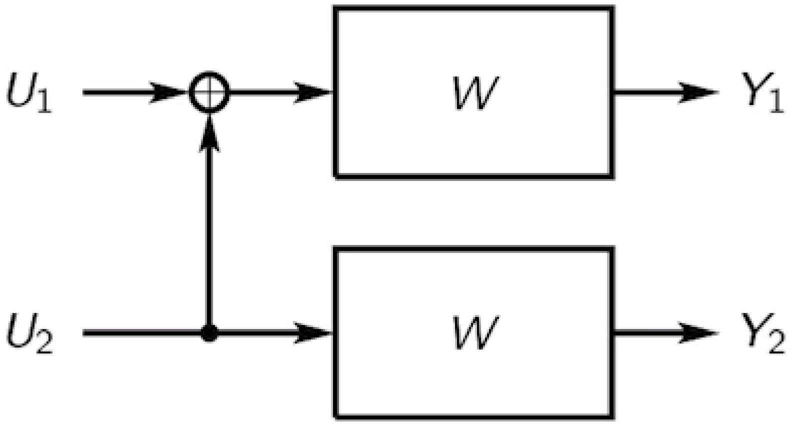


Two independent channels

Now consider two inputs U1 and U2 that feed into the X1 and X2 inputs. The connections are as made as shown below.

$$X_1 = U_1 \oplus U_2$$
$$X_2 = U_2$$

This results in the network shown below. We will see later that this simple network polarizes the output.



Connecting the two-channel as shown above polarizes the output

Binary erasure channel

Now let's define W before we proceed further. Let's model **W** to the Binary Erasure Channel (BEC). BEC is defined as:

- X is unsuccessfully decoded with a probability **p**. In the BEC, the bit is erased, so the receiver is able to detect that a bit has been lost.
- X is successfully decoded into Y with a probability **1-p**.

Now let's analyze the network we discussed above in terms of two bit-channels:

- $\mathbf{W}^-$: U1 → (Y1, Y2)
- $\mathbf{W}^+$: U2 →(Y1, Y2)

W^{-}: U_1 → (Y_1, Y_2)

The **W⁻** channel tries to reconstruct the input received on the U1 channel. The following possibilities exist when the W channel is modeled as a BEC:

- Both channels are successfully decoded — probability **$(1\text{-}p)^2$**
- The first channel is erased but the second one is decoded successfully — probability **p(1-p)**
- The first channel is decoded successfully but the second channel is erased — probability **(1-p)p**
- Both channels are erased — probability **p^2**

Polarizing channel set

These possibilities are summarized below:

Here U1 can be successfully decoded only in the first case as U1 can be extracted from U1⊕U2 as U2 was also decoded successfully. All other cases represent decode failure as it is not possible to extract U1.

We have created a **W⁻** with the following characteristics:

- U1 is successfully decoded with the probability $\mathbf{(1\text{-}p)^2}$
- U1 decode fails with the probability of $\mathbf{p(1\text{-}p) + (1\text{-}p)p + p^2 = 2p\text{-}p^2}$

Note here that we have managed to create a worse channel than the BEC. If the BEC had a 30% chance or erasure:

- BEC Success probability is 0.7
- BEC Erasure probability is 0.3

For the **W⁻** bit-channel:

- **W⁻** success probability is $(1.0\text{-}0.3)^2 = 0.7^2 =$ **0.49**
- **W⁻** erasure probability is $2{*}0.3\text{–}0.3^2 = 0.6\text{–}0.09 =$ **0.51**

W^+: $U_2 \rightarrow (Y_1, Y_2)$

Now let's turn our attention to the **W⁺** channel. The channel reconstructs the input received on the U2 channel. With a BEC channel, we have the following possibilities:

- Both channels are successfully decoded — probability $\mathbf{(1\text{-}p)^2}$
- The first channel is erased but the second one is decoded successfully — probability **p(1-p)**
- The first channel is decoded successfully but the second channel is erased — probability **(1-p)p**
- Both channels are erased — probability $\mathbf{p^2}$

Polarizing channel set

These possibilities are summarized below. :

> ***Note:*** *From the previous section we know that* $\mathbf{W^-}$ *is a worse channel that the BEC. If we send a known value, this can help in improving the decode for the* $\mathbf{W^+}$ *channel. This is part of our plan to polarize an OK channel into a good and bad bit-channels. We will send our data bits on the good bit-channel and clamp the bad bit-channel to a known value.*

Here U2 can be successfully decoded in the first and second cases as U2 was received successfully. In the third case, U2 can be extracted from U1$\oplus$U2 as we have advance knowledge of U1. The last case represents a decode failure as it is not possible to extract U2.

We have created a $\mathbf{W^+}$ with the following characteristics:

- U1 is successfully decoded with the probability $\mathbf{(1-p)^2 + p(1-p) + (1-p)p = 1-p^2}$
- U1 decode fails with the probability of $\mathbf{p^2}$

Note here that we have managed to create a better channel than the BEC. If the BEC had a 30% chance or erasure:

- BEC Success probability is 0.7
- BEC Erasure probability is 0.3

For the $\mathbf{W^+}$ bit-channel:

- $\mathbf{W^+}$ Success probability is $1.0–0.3^2 = 1–0.03=$ **0.91**
- $\mathbf{W^+}$ Erasure probability is $0.3^2 =$ **0.09**

Experimenting with polarization

We saw above that when the polarizing transform was applied to a BEC with an erasure probability of p, we ended up with two bit-channels:

- **W^+** with an erasure probability of **p^2**
- **W^-** with an erasure probability of **$2p-p^2 = p(2-p)$**

> *The results above clearly show that **W^+** has a lower erasure probability than **W^-**. Since $0 \leq p \leq 1$ it follows that **$p^2 \leq p(2-p)$***

Now, let's run a little experiment to understand the power of the polarizing transform.

> ***Note:** You will need a Linux, macOS or Windows 10 based machine to run the experiment. On Windows 10 you will need the Windows Subsystem for Linux (WSL). On all platforms, you will be using awk and gnuplot.*

polarize script file

In your favorite editor create the following script file that accepts an erasure probability input as column 1 `$1` and prints the erasure probability of the **W^+** `p*p` and **W^-** `p*(2-p)` bit-channels. The file is named `polarize`.

```
awk '{p = $1; print p*p; print p*(2-p);}'
```

polarize once

We run the `polarize` script with the erasure probability of 0.3. The script returns the erasure probability of **W^+** (0.09) and **W^-** (0.51).

```
$ chmod +x ./polarize
$ echo 0.3 | ./polarize
0.09
0.51
```

The polarize script can be visualized as shown below.

Visualization of the "polarize" script

polarize twice

We have built a polarizing building block that takes two OK bit-channels and procedures a good (**W^{+}**) and bad (**W^{-}**) bit-channel. Let's cascade this building block as shown below.

A four-channel polarizer

The cascade results in the following channels:

- **$W^{-\,-}$: V1 → Y1Y2Y3Y4**
- **$W^{-\,+}$: V2 → Y1Y2Y3Y4V1**
- **$W^{+\,-}$: V3 → Y1Y2U1Y3Y4U3**
- **$W^{+\,+}$: V4 → Y1Y2U1Y3Y4U3V3**

We can represent this cascade structure by just invoking the polarize script as shown below:

```
$ echo 0.3 | ./polarize | ./polarize
0.0081
0.1719
0.2601
0.7599
```

The cascade of two polarize script calls is visualized below. Here we that the original channel with erasure probability 0.3 got converted into two bit-channels of erasure probability 0.09 and 0.51. Each of these outputs is further fed into the next polarize script. The awk script operates on the two rows and we get the final four bit-channels.

The four bit-channels have been colored to show the impact of the two-step polarization. We have produced a spectrum of bit-channels: great, good, bad and ugly.

Visualization of a cascade of two "polarize" scripts. We see in the final output the polarization into good (green) and bad channels (orange and red)

polarize four times

We can extend this further with a polarize4 script that just piping the polarize script four times.

polarize4 file

```
./polarize | ./polarize | ./polarize | ./polarize
```

Piping in the erasure probability of 0.3 via polarize4 produces a total of 16 bit-channels. We sort the output to clearly see the great channels we have produced in this process.

```
$ echo 0.3 | ./polarize4 | sort -g
4.30467e-09
0.000131216
0.000260319
0.000873179
0.00457679
0.0320085
0.058226
0.0987531
0.130727
0.2048
0.333446
0.529747
0.700296
0.82145
0.888027
0.996677
```

plot

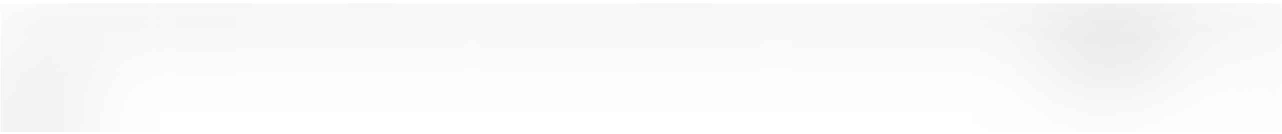
Let's save the output in a file…

```
$ echo 0.3 | ./polarize4 | sort -g > 4level
```

…and then plot it using gnuplot.

```
$ gnuplot
gnuplot> plot "4level"
```

The output plot clearly shows the results of the four-level polarization. The good bit-channels are shown with an erasure probability that is close to 0. There is no free lunch, so we do end up with some really bad channels with erasure probability that is close to 1.



A plot of channels (X-axis) ordered by erasure probability (Y-axis) with a 4-level polarization. Polarization has resulted in 16 bit-channels. Channels 1 to 4 are practically error-free.

8-level polarization

At an 8-level polarization, we get a total of 256 bit-channels. We are getting to a point where 70% of bit-channels are great and 30% of the bit-channels are bad.

```
$ echo 0.3 | ./polarize4 | ./polarize4 | sort -g > 8level
```

8-level ”polarization” results in 256 bit-channels. The polarization has resulted in ~70% of the channels being great for data transmission.

12-level polarization

The 12-level we see even sharper polarization. The plotline from good channels to bad channels is tending to be vertical, this means that most channels have moved to the extremes: great and ugly.

```
echo 0.3 | ./polarize4 | ./polarize4| ./polarize4 | sort -g >
12level
```

A 12-level polarize further solidifies the 70% great and 30% bad channel split

16-level polarization

We combine the polarize4 scripts into a 16 level script.

polarize16 file

```
./polarize4 | ./polarize4 | ./polarize4 | ./polarize4
```

Running the 16-level runs shows a very high level of polarization. We have 70% great channels with close to 0 erasure probability. 30% of the channels are complete duds with an erasure probability of 1.

```
$ echo 0.3 | ./polarize16 | sort -g > 16level
```

A 16-level polarization of a BEC with 30% erasure probability has created a set of bit-channels where 70% of the channels are virtually error-free. The remaining 30% of the channels are terrible.

Learn more

Most of the material presented above is based on the excellent presentation on polar codes from Professor Emre Telatar.

5g Polarization Polar Codes Signal Processing